フロンティア法から生成されるZDDの幅解析 (理論計算機科学の新展開)

AUTHOR(S):

高野, 圭司

CITATION:

高野, 圭司. フロンティア法から生成されるZDDの幅解析 (理論計算機科学の新展開). 数理解析研究所講究録 2013, 1849: 77-82

ISSUE DATE:

2013-08

URL:

http://hdl.handle.net/2433/195106

数理解析研究所講究録
第 1849 巻 2013 年 77-82
2012 年度冬の LA シンポジウム [S6]

フロンティア法から生成される ZDD の幅解析

東京工業大学　高野 圭司
Keiji Takano, Tokyo Institute of Technology

1　はじめに

Knuth が提唱した，データ構造 ZDD を用いた Simpath アルゴリズムにより，非常に高速なグラフ上の s-t パス全列挙が可能になった [5]．さらに，他の部分グラフを列挙できるよう，Simpath を拡張したフロンティア法が提唱された [7]．しかし，このアルゴリズムは一般の入力グラフについては頂点数，辺数といったサイズに対して指数時間かかることが実験的に確かめられていた．本研究では列挙対象をサイクル，s-t パスとし，入力グラフについては完全グラフ，平面グラフ，グリッドグラフの 3 つを対象に，ZDD 構築に必要な時間，領域計算量について [4, 8] を下に理論的な解析を行った．計算量の解析にあたっては ZDD の最大幅が関係しており，本研究では 3 種類のグラフから生成される ZDD の最大幅が，それぞれマッチング多項式，Catalan number，Motzkin number などの，組合せ論でしばしば表れる数列と関連していることを示した．

2　ZDD

ZDD (Zero-suppressed BDD; ゼロサプレス型 BDD) [6] は非循環有向グラフにより，条件を満たす組合せ集合の集合を圧縮索引化した状態で保持したデータ構造である．用語の混乱を避けるため，ZDD を表すグラフにおいて，頂点を節点，辺を枝と呼ぶことにする．節点は組合せ集合における要素に対応している．節点からはその要素が含まれないことを表す 0-枝と，含まれることを表す 1-枝が伸びている．また，最終的な結果を表す特殊な節点が 2 つあり，条件を満たさないという結果を表す 0-終端節点と，満たすという結果を表す 1-終端節点がある．この 2 つの節点からは枝は伸びていない．ZDD ではある

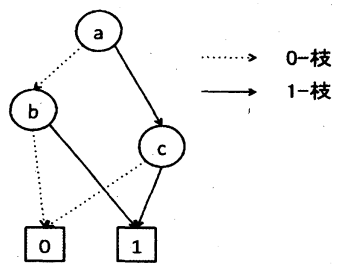

図 1: 組合せ集合 $\{\{ac\},\{b\}\}$ を保持した ZDD

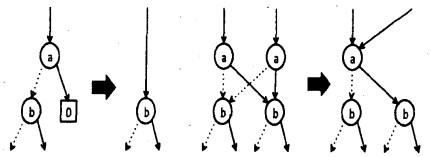

図 2: ZDD の削除規則と共有規則

節点の 1-枝が 0-終端節点を直接指しているときに，その節点を取り除く．また，同じラベルが付いた 2 つの節点について，2 つの 0-枝，1-枝の指す先がそれぞれ同じである場合，その 2 つの節点を 1 つにまとめることができる．ZDD はデータを圧縮索引化して保持するほか，構築後の演算処理が容易かつ効率的であることが特徴である．構築された ZDD に演算処理を施せば，さらに条件を絞った解を保持した ZDD を再構築できる．

3 フロンティア法

本章では s-t パスの全列挙に話を絞ってこのアルゴリズムの説明を行う [5, 7]. G 上の s-t パスとは, G の部分グラフ $\acute{G} = (\{\acute{v}_1, \cdots, \acute{v}_{l+1}\}, \{\acute{e}_1, \cdots, \acute{e}_l\})$ である. ただし, 任意の i について $\acute{e}_i = (\acute{v}_i, \acute{v}_{i+1}) \in E$, $\acute{v}_i \in V$, $\acute{v}_1 = s$, $\acute{v}_{l+1} = t$ である. 以降は辺の集合のみをパスとみなす. フロンティア法は図 3 のように, 全ての s-t パスを表す辺の集合族を $e_1, \cdots, e_m$ を変数とする非既約な ZDD で出力する. 変数 (辺) 処理順序については s に付随する辺から何らかの手順で変数順が $e_1, \cdots, e_m$ と決められているとする.

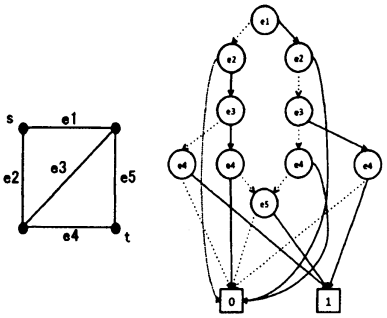

図 3: G とその s-t パス全てを保持した ZDD

フロンティア法は, 頂上の節点からトップダウンで ZDD の枝と節点を段階的に作っていく. 節点 n の生成前に, 枝刈り (0,1-終端節点直結) ができるかを判定し, 可能ならばそれを行う. また n の生成時に, 既に生成した節点 $\acute{n}$ と共有が可能かを判定し, 共有可能ならば n を生成せずに $\acute{n}$ を接続先とする. 共有, 枝刈りを行うために, フロンティア, mate 配列と呼ばれる概念を導入する. フロンティア F_i とは, i 番目までの辺が処理されたときに, 処理済み辺 (辺を含める, 含めないを既に決定した辺) と未処理辺 (辺を含める, 含めないをまだ決定していない辺) の両方に接している頂点集合のことで, $F_i = \{u \in V | \exists v \in V, \exists w \in V, (u,v) \in \cup_{j=1}^{i} e_j, (u,w) \in \cup_{j=i+1}^{m} e_j\}$ で定義される. ただし, $F_0 = F_m = \phi$ とする. mate 配列とは ZDD の各節点に持たせる配列であり, 配列の長さはの最大値は $|V|$ に等しい. mate の値は $v, w \in V$ について,

$$mate[v] = \begin{cases} w & (v,w \text{が部分パス } v-w \text{ を構成}) \\ v & (v \text{は孤立点}) \\ 0 & (v \text{はある部分パスの途中点}) \end{cases}$$

と定義する. mate の初期値は s, t でない各 v について $mate[v] = v$ とし, $mate[s] = t$, $mate[t] = s$ とする. これは辺 (s,t) をダミー辺として加え, 辺 (s,t) を含むサイクルを列挙する問題に置換したと考えることもできる. 実用上は F_{i-1} 上の頂点の mate 値のみ記憶させる. これは共有判定の際, F_{i-1} の mate 値しか利用しないためである.

図 4 は共に辺 (6,9) を次に処理する状態であるとする. 処理する図ではどちらも 1 番と 8 番の頂点, 6 番と 7 番の頂点が処理済み辺が構成する部分パスの両端になっているため, 今後追加して s-t パスとなる辺の組合せは同じである. よって未処理辺を処理する上で必要な情報は, フロンティア上の頂点 (図では 6 番, 7 番, 8 番の頂点) がその状態においてパスの端, 途中点あるいは孤立点のいずれであるか, パスの端であるならばもう一方の端はどの頂点かだけである. それらの状態を mate に置き換えると, 図 4 では 2 つの状態でいずれも $mate[6] = 7$, $mate[7] = 6$, $mate[8] = 13$ となり, 共有が可能となる.

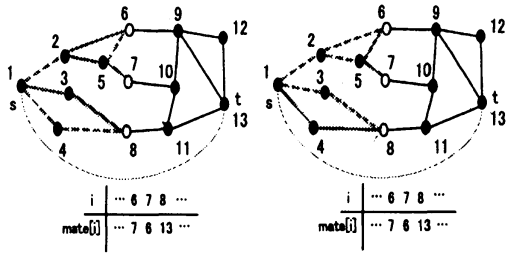

図 4: 共有が可能な 2 つの状態

フロンティア法の疑似コードは Algorithm 1 のようになる. 5 から 17 行目で節点 n の 0,1-枝の先の節点を生成している. 6 行目の CheckTerminal(n,i,x) は, n の x-枝の先が 0,1-終端節点となるかの判定を

Algorithm 1 $(G=(V,E),s,t)$

```
N_1 ← {e_1}
N_i ← ∅ for i = 2,···,m+1
for i = 1 to m do
  for all n ∈ N_i do
    for all x ∈ {0,1} do
      ń ← CheckTerminal(n,i,x) // 0-終端節点, 1-終端節点または nil を返す
      if ń = nil then
        新しい節点を生成し，それを ń に代入
        ń.mate ← UpdateMate(n.mate,i,x)
        if ń と共有可能な n̂ ∈ N_i が存在 then
          ń←n̂
        else
          N_{i+1}← N_{i+1} ∪ {ń}
        end if
      end if
      n の x 枝の先に ń を接続
    end for
  end for
end for
```

行っている．$e_i=(v,w)$ を加えようとしたときに以下の条件が成立した場合，0,1-終端節点を返す．

(A)$mate[v]=0$ または $mate[w]=0$ のとき
既にある部分パスに分岐が生じてしまうので，0-終端節点に直結させる．

(B)$mate[v]=w$ のとき
このときサイクルが完成するが，v,w 以外のあるフロンティア頂点 $u\in F_i$ について $mate[u]\neq 0$ かつ $mate[u]\neq u$ の場合，u はあるパスの端点であり，サイクルに余分な辺があるため 0-終端節点に直結させる．そうでない場合は 1-終端節点に直結させる．

(C)mate を更新後，$\exists u\in F_i\backslash F_{i+1}$ で $mate[u]\neq 0$ かつ $mate[u]\neq u$ のとき
フロンティアから抜ける頂点 u が部分パスの端点である場合，u から先にパスを伸ばせなくなるため，0-終端節点に直結させる．

CheckTerminal(n,i,x) が nil を返した場合には 7 行目以降が実行され，$\acute{n}$ が新しく生成される．8 行目の UpdateMate$(mate,i,x)$ は mate の更新を行う．辺集合に $e_i=(v,w)$ を含める（1-枝の先の節点を生成する）場合には，$x:=mate[v], y:=mate[w]$ とすると，x-y パスが新たに生じるので，$mate[x]:=y, mate[y]:=x$ と変更する．さらに $v\neq x$ のとき $mate[v]:=0$ とし，$w\neq y$ のとき $mate[w]:=0$ とする．10 行目では 2 つの節点 $\acute{n},\hat{n}$ の共有判定を行っている．$\acute{n},\hat{n}$ について，フロンティア上の同じ頂点の mate 値をそれぞれ調べる．全ての値が一致していれば，n から新しい節点を生成せず，n の x-枝の先を $\hat{n}$ に繋ぐ．

4 フロンティア法の計算量

フロンティア法の計算量 [8] の主要部分を占めているのは UpdateMate, CheckTerminal，および節点の共有判定である．UpdateMate については mate 配列の長さに比例した時間がかかり，$O(\max|F_i|)$ で更新できる．CheckTerminal についても UpdateMate と枝刈り判定が計算時間の主要部分を占め，これらを合計すると $O(\max|F_i|)$ である．節点の共有判定についても，ハッシュ表を用いることで，作業領域が十分に大きければ $O(\max|F_i|)$ でできる．よって，1 つの節点を生成するのに $O(\max|F_i|)$ だけの時間をかければ良い．よって Algorithm 1 より，フロンティア法の時間計算量は $O(m\max|N_i|\max|F_i|)$ である．領域計算量については，$O(\max|N_i|)$ だけの作業領域が必要である．ZDD 構築の計算量を解析するにあたっては，$\max|F_i|$, $\max|N_i|$，すなわちフロンティアの最大サイズ，および ZDD の最大幅が重要な要素となる．本研究ではグラフクラスと辺処理順序を固定し，$\max|F_i|$ が明白になっている下で $\max|N_i|$ の理論的な解析を行った．ただし，これらの値については用いるハッシュ関数によって値が変わる可能性があるため，完全ハッシュ関数を用いたという仮定を置いた．詳しい結果については次章以降で述べる．

5 ZDD 最大幅の解析

5.1 完全グラフの場合

完全グラフにおいてはマッチング多項式 [1]

$$|M_{K_n}(x)| = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \frac{n!}{(n-2k)!k!2^k} x^{n-2k}$$

を用いることで，最大幅の上界を簡潔な形で表せる．$|M_{K_n}(2)|$ は「円周上に並んだ n 個の点のうち，$2k$ 個の点を選んで k 本の弦で結び，残りの $n-2k$ 個の点を 2 色に塗り分ける場合の数」と対応している．また，$|M_{k_n}(2)| \leq (\sqrt{3})^n n!!$ である．本論文で用いた辺処理順序について述べる．各頂点に任意に番号を振り，辺に番号対 (u,v) $(u<v)$ を定める．この対が辞書式順序になるよう処理順を定める．例えば，4 頂点完全グラフ K_4 であれば，(v_1,v_2), (v_1,v_3), (v_1,v_4), (v_2,v_3), (v_2,v_4), (v_3,v_4) の順で処理を行う．この順序を canonical edge ordering と呼ぶことにする．なお s-t パス列挙においては，始点を v_1，終点を v_2 とした．以降，示した結果の証明については紙面の都合により，サイクルのみ証明を行う．

定理 1. canonical edge ordering を用いた完全グラフ K_n にフロンティア法を用いてサイクル，$s-t$ パスを全列挙した場合，ZDD 最大幅の上界はそれぞれ $|M_{K_{n-1}}(2)| - 2^{n-1} + 1$，$|M_{K_{n-2}}(2)| - 2^{n-2}$ である．

証明. フロンティア F 上の頂点はパス端点，パス途中点，孤立点のいずれかであることに着目し，mate 配列のとり得る種類数を評価する．辺 (v_1, v_{n-1}) を処理した直後は $F = \{v_2, \cdots, v_n\}$ である．このとき $n-1$ 個の頂点のうち，パス端点対が k 個 $(1 \leq k \leq \lfloor \frac{n-1}{2} \rfloor)$ 存在する場合，残りの $n-1-2k$ 個の頂点は孤立点または途中点であり，考え得る種類数の上界は 2^{n-1-2k} 通りである．$n-1$ 個の頂点全てが孤立点である状態も考慮し，種類数総計の上界を式で表すと，

$$\begin{aligned} & 1 + \sum_{k=1}^{\lfloor \frac{n-1}{2} \rfloor} \frac{\binom{n-1}{2} \cdots \binom{n-1-2k+2}{2}}{k!} 2^{n-1-2k} \\ = \ & |M_{K_{n-1}}(2)| - 2^{n-1} + 1 \end{aligned}$$

となる． □

5.2 平面グラフの場合

平面グラフの解析では平面性を用いることで，ZDD 最大幅についてより細かい解析を行うことができる．本研究では Catalan number [2]

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

を用いた最大幅の上界を示した．C_n については $C_n \approx n^{-\frac{3}{2}} 4^n$ であることが知られている．以下ではまず，サイクル，s-t パスを列挙するのに用いた辺処理順序の定義について述べる．

定義 2. 連結な単純平面グラフにおいて，以下の条件を常に満たす辺処理順序を *L-ordering* という．

- 処理済み辺で構成される部分グラフ $\acute{G} = (\acute{V}, \acute{E})$ において，フロンティアに属する頂点集合を F とおく．このとき，$\acute{V} \setminus F$ が閉曲線 L の内 (外) 部領域に含まれ，F が L 上となるよう L を定められる．

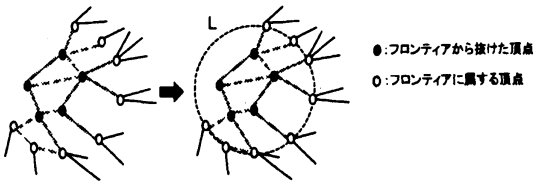

図 5: *L-ordering* による頂点分割問題への帰着

定義 3. 連結な単純平面グラフにおいて s-t パスを列挙するとき，以下の条件を常に満たす辺処理順序を *Ĺ-ordering* という．

- 始点 s に付随する辺を始めに全て処理した，処理済み辺で構成される部分グラフ $\acute{G} = (\acute{V}, \acute{E})$ において，フロンティアに属する頂点集合を F とおく．このとき，始点 s が閉曲線 L 上の外 (内) 部領域，$\acute{V} \setminus F$ が L の内 (外) 部領域，F が L 上となるよう L を定められる．

これらの定義を導入すれば，幅解析の問題を円周上の頂点における頂点分割の問題に帰着させることができる．ジョルダンの閉曲線定理より，平面に閉曲線を 1 つ定めると，その閉曲線によって平面を内部と外部の 2 つの領域に分けられる．$\acute{V} \setminus F$ が，生成された領域のいずれか 1 つに含まれるというのが，上記の定義で定められた条件である．一般性を失わないため，以降 $\acute{V} \setminus F$ は内部領域に含まれるものとする．*L-ordering*，*Ĺ-ordering* に該当する順序の 1 つとして，幅優先による辺処理順序がある．これはまず，任意に指定した頂点（始点）から幅優先探索で順に頂点に番号を振り，canonical edge ordering で辺に順序を定めた方法である．上記の辺順序で，平面グラフの最大幅について以下の定理を示した．

定理 4. 連結な単純平面グラフにおいて，*L-ordering* でサイクルを全列挙したときを考える．フロンティア最大サイズが k のとき，ZDD 最大幅の上界は C_{k+1} である．

証明. *L-ordering* によって辺を処理した場合，L の外部領域の辺は全て未処理辺であるため，フロンティアに属する頂点は全て L の内部領域にある辺でのみ他のフロンティア頂点と繋がることができる．いま，フロンティアのサイズが k であるときを考える．k 個のフロンティア頂点のうち $2i$ 個のフロンティア頂点がパス端点であるとき，この $2i$ 個の繋ぎ方の総数は，円周上の $2i$ 個の点全てを i 本の弦で互いに非交差に結ぶ場合の数，すなわち C_i で抑えられる．残りの $k-2i$ 個の点は孤立点またはパス途中点であるため，残りの $k-2i$ 個のフロンティア頂点における mate 配列の種類数を高々 2^{k-2i} 通りで抑えられる．よって，k 個の頂点の繋ぎ方は $\sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} \binom{k}{2i} C_i 2^{k-2i}$ 通りで抑えられるが，これは Touchard's Catalan Identity [3] により，

$$\sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} \binom{k}{2i} C_i 2^{k-2i} = C_{k+1}$$

と変形できる． □

同様の手法で s-t パスについても以下の定理を示した．

定理 5. 連結な単純平面グラフにおいて，*Ĺ-ordering* で s-t パスを全列挙したときを考える．$s \notin F$ でフロンティア最大サイズが k のとき，ZDD 最大幅の上界は $C_{k+2} - 2C_{k+1}$ である．

5.3 グリッドグラフの場合

グリッドグラフにおいては，Motzkin number [2]

$$M_n = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2k} C_k$$

および Generalized ballot number

$$B_n = M_{n+1} - M_n = \sum_{k=0}^{n-1} M_k M_{n-1-k}$$

を用いた最大幅の真の値を示した．M_n は「円周上に並んだ n 個の点のうち，$2k$ 個の点を選んで k 本の互いに非交差な弦で結ぶ場合の数」と対応している．M_n, B_n の値については $M_n \leq 3^n$, $B_n \leq 2 \cdot 3^n$ であることが知られている．グリッドグラフでは行優先による辺処理順序を定めることで，ZDD の最大幅をさらに抑えられる．行優先による順序とは，k^2 個の頂点に対して，左上から右下へ行優先で頂点に $v_1, v_2, \cdots, v_{k^2}$ とつけ，この順序で辺に canonical edge ordering を行った順序と定義する．なお s-t パスにおける列挙においては，左上の頂点 v_1 を始点，右下の頂点 v_{k^2} を終点とした．$k \times k$ 頂点グリッドグラフ $L_{k,k}$ における ZDD 最大幅の真の値について以下の定理を示した．

定理 6. グリッドグラフ $L_{k,k}$ に対して，行優先で辺処理順序を定めサイクル，s-t パスを列挙したとき，ZDD 最大幅はそれぞれ $M_k+2M_{k-1}-2$, B_k+2B_{k-1} である．

証明. $j(2k-1)+l+1$ 番目 ($\lceil \frac{k}{2} \rceil \leq j \leq k-2, l = 3, 5, \cdots, 2k-3$) の辺を処理する直前が ZDD 幅が最大になるので，このとき存在する k 個のフロンティア頂点に対する mate 配列の種類数を考える．k 個のフロンティア頂点について，小さい番号から順に

$v_p, v_{p+1}, \ldots, v_{p+k-1}$ とする．k 個の頂点の処理済み辺による繋がり方によって以下の場合に分けられる．

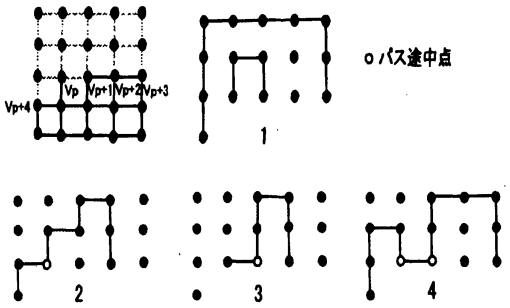

図 6: 処理済み辺の繋がり方による場合分け

1. パス途中点がない
2. v_p がパスの途中点
3. v_p と v_{p+1} が辺で繋がり，v_{p+1} がパスの途中点で v_p がパスの端点
4. v_p と v_{p+1} が辺で繋がり，v_p, v_{p+1} がパスの途中点

1, 2, 3, 4 の状態数はそれぞれ M_k 通り，$M_{k-1}-1$ 通り，$M_{k-1}-M_{k-2}$ 通り，$M_{k-2}-1$ 通りであり，その和は $M_k+2M_{k-1}-2$ 通りとなる．□

n	最大幅	幅上界	合計節点数	全サイクル数
3	2	2	7	1
4	6	7	21	7
5	18	28	69	37
6	47	111	231	197
7	105	436	758	1172
8	313	1723	2405	8018
9	847	6938	7419	62914
10	2042	28675	22471	556014
11	4837	122086	67323	5488059
12	14233	536031	200722	59740609
13	39101	2426259	598582	710771275

表 1: K_n の全サイクルを保持する ZDD のサイズ

k	最大幅	合計節点数	全サイクル数
3	6	47	13
4	15	220	213
5	37	910	9349
6	91	3444	1222363
7	227	12325	487150371
8	575	42514	603841648931
9	1479	143072	2318527339461265
10	3856	473105	27359264067916806101
11	10172	1544265	9888088110462835950680 99

表 2: $L_{k,k}$ の全サイクルを保持する ZDD のサイズ

完全グラフ K_n とグリッドグラフ $L_{k,k}$ でサイクル列挙の数値実験を行ったところ，表 1,2 のようになった．今後の課題として，完全グラフ上界のより厳密な解析，他のグラフクラスにおける同様の解析などが挙げられる．後者の課題のためには [8, 9] より，グラフのパス幅や混合探索数を求めるアルゴリズムの研究が不可欠になるのではないかと考えている．

参考文献

[1] I. Gutman and H. Hosoya. On the calculation of the acyclic polynomial. *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)*, 48(4):279–286, 1978.

[2] F.R. Bernhart. Catalan, motzkin, and riordan numbers. *Discrete Mathematics*, 204(1):73–112, 1999.

[3] L.W. Shapiro. A short proof of an identity of touchard's concerning catalan numbers. *Journal of Combinatorial Theory, Series A*, 20(3):375–376, 1976.

[4] K. Sekine, H. Imai, and S. Tani. Computing the Tutte polynomial of a graph of moderate size. *Algorithms and Computations*, pages 224–233, 1995.

[5] D. E. Knuth. The art of computer programming: Bitwise tricks & techniques; binary decision diagrams, volume 4, fascicle 1, 2009.

[6] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Design Automation, 1993. 30th Conference on*, pages 272–277. IEEE, 1993.

[7] 川原純，湊真一．組合せ問題の解を列挙索引化する ZDD 構築アルゴリズムの汎用化．(Theoretical Foundations of Computing)．**電子情報通信学会技術研究報告：信学技報**，112(93):1–7, 2012.

[8] 斎藤寿樹．フロンティア法の計算量について．2011 **年度** ERATO **湧離散構造処理系プロジェクト講究録**, 2012.

[9] 内海友雄，今井桂子．混合探索による順序を用いた Jones 多項式の計算手法．**情報処理学会研究報告．*AL*，アルゴリズム研究会報告**, 2004(34):95–100, 2004.